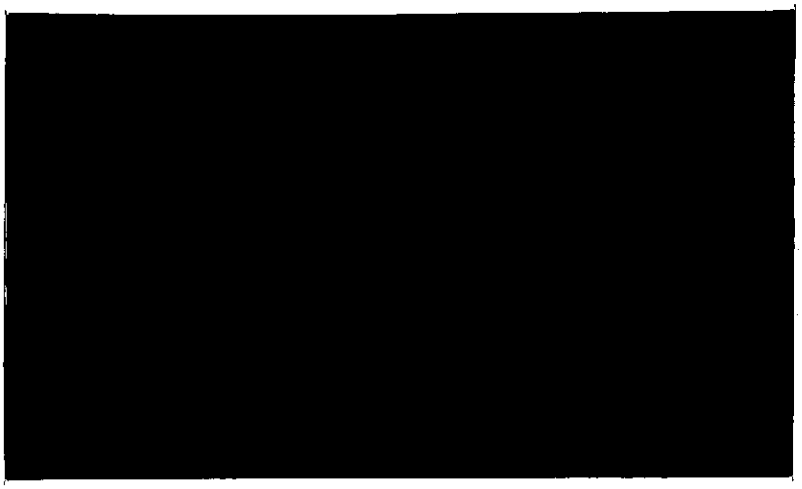
LOGIC OF COMPUTERS GROUP



Department of Computer and Communication Sciences
2080 Frieze Building
The University of Michigan
Ann Arbor, Michigan 48104

THE UNIVERSITY OF MICHIGAN
COLLEGE OF LITERATURE, SCIENCE, AND THE ARTS
Computer and Communication Sciences Department

CONVERGENCE PROPERTIES OF SIMPLE GENETIC ALGORITHMS

by

Albert D. Bethke
Bernard P. Zeigler
David M. Strauss

July 1974

Technical Report No. 159

with assistance from

National Aeronautics
and Space Administration
Grant No. NGR-23-005-602
Washington, D.C.

I. BACKGROUND AND INTRODUCTION

The long range goal of our research project has been to answer the following questions:

1) How broad is the domain of useful application of genetic direct search algorithms (employing techniques suggested by natural adaptive systems)?
2) Should one general form of algorithm be applied in all situations or are different configurations or parameter settings appropriate in different environments?

Our previous research [see Bosworth 1972, Foo 1972, and Zeigler 1973] was devoted to answering the more particular question: How well could genetic algorithms preform in comparison to standard direct search techniques currently available? We studied the cases of unimodal, multimodal, and noisy functions. Also, several versions of genetic optimization algorithms were constructed during the exploratory phase of our investigation. Many variations were incorporated and tested in the possibility that they could prove essential to algorithm performance. As could be expected, this led eventually to a complex software package.

The question which we studied during the final year of our research was: What are the essential parameters determining the behavior of genetic algorithms and what values should be assigned to these parameters for optimum performance? So, during the final phase of the project the algorithm was refashioned into a much more simple and elegant form. This final version contains only the basic features which our experimentation with earlier versions suggested to be essential. As a consequence, there were only 8 parameters which could be varied to control the performance of the final optimization program.

Having thus reduced the parameter space to an (almost) manageable size, we made a large number of computer runs while systematically varying the parameter values. Most of the results stated later in this report are based on the progress curves (function value of the best point versus the number of function evaluations) obtained from these runs. Other results are based on the variability of the population as the run progresses.

II. ALGORITHM DESCRIPTIONS

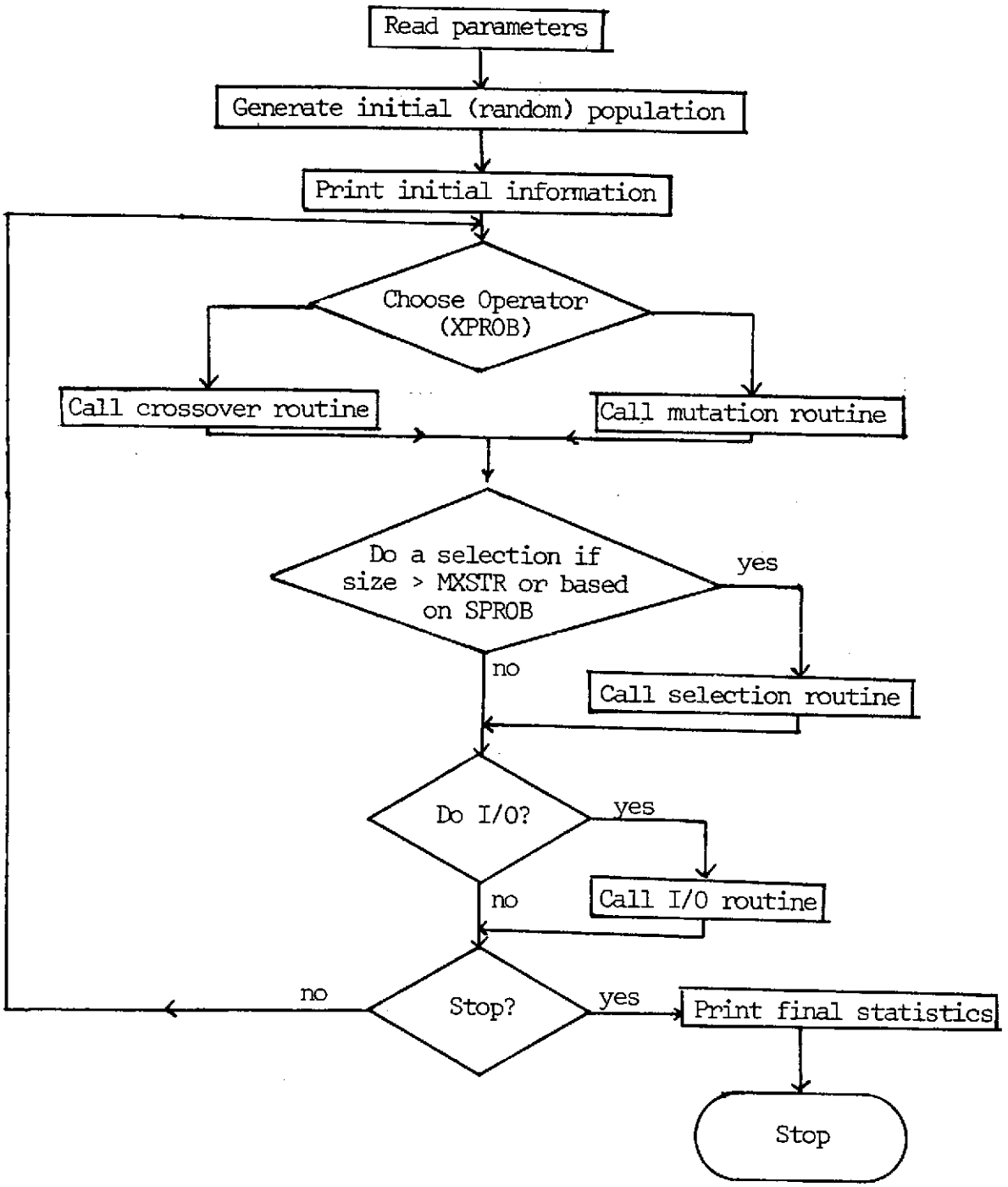
Our optimization algorithm belongs to a class of algorithms known as genetic algorithms. Instead of sampling several points about a base point to compute an approximate gradient and follow it, genetic algorithms explore the space by searching out the best hyperplanes using a "population" of many points and the genetic operators crossover and mutation and possibly others. The basic paradigm goes like this:

0. Generate an initial population of strings (using a random number generator) -- each string (also called an individual) represents a point in the objective function's domain.
1. Use genetic operators to produce new individuals and add them to the population. Usually the strings chosen to be operated on (crossed over or mutated or whatever) are chosen randomly but the choice is biased according to the utility (function value) of each individual. That is, the best string has the highest probability of being chosen to participate while the worst string is least likely to be used.
2. Use some selection routine to reduce the size of the population back to the initial size. Generally the decision of which strings to keep is also biased according to the function values.
3. Repeat steps 1 and 2 until some termination criterion is met.

In our particular algorithm, the basic activity cycle consists of the following:

1. Decide whether to use crossover or mutation this time. (The parameter XPROB is the probability of doing crossover, and 1-XPROB is the probability of mutation.)
2. Perform the chosen operation.

MAIN PROGRAM



3. Decide whether or not to reduce the population down to the initial size and carry out the selection if that is the decision. (The parameter SPROB is the probability of using the selection routine at this point in the cycle.) If the population has reached or exceeded the maximum size allowed for this run (MXSTR), then selection must be performed. (The initial size of the population is given by the parameter NSTR.)

As noted above, each string represents a point in the objective function's domain. However, genetic algorithms are best suited to working on strings with only a small number of possible values (alleles) at each position. That is, if the domain space consists of roughly 1000 points, it would be better to represent these points using "binary strings" of length 10 than by using "decimal strings" of length 3. Since we are working with functions of several real variables, we construct a uniform grid on the domain space and each coordinate of the function is allowed to take on only a small number of values -- the parameter NALEL in our programs gives this number. (The grid could be refined by using 2 or more positions per coordinate to encode the domain points as strings.)

The range of values allowed along each dimension in the domain space was limited according to which function was being optimized. For the "standard" function, $\sum_i ix_i$, each coordinate was restricted to the range 0 to 4. This range was partitioned into n (the value of NALEL) equal segments and the different alleles were associated with the midpoint of each segment. Thus, if there are 2 possible values for each position in a string, they correspond to x_i values of 1 and 3. Similarly, if there are 4 alleles, then the possible x_i values are .5, 1.5, 2.5, and 3.5. For Wood's function, the domain was a hypercube with a range of -2 to 2 in each of the 4 dimensions. Thus an individual is a string of integers each of which lies between 0 and

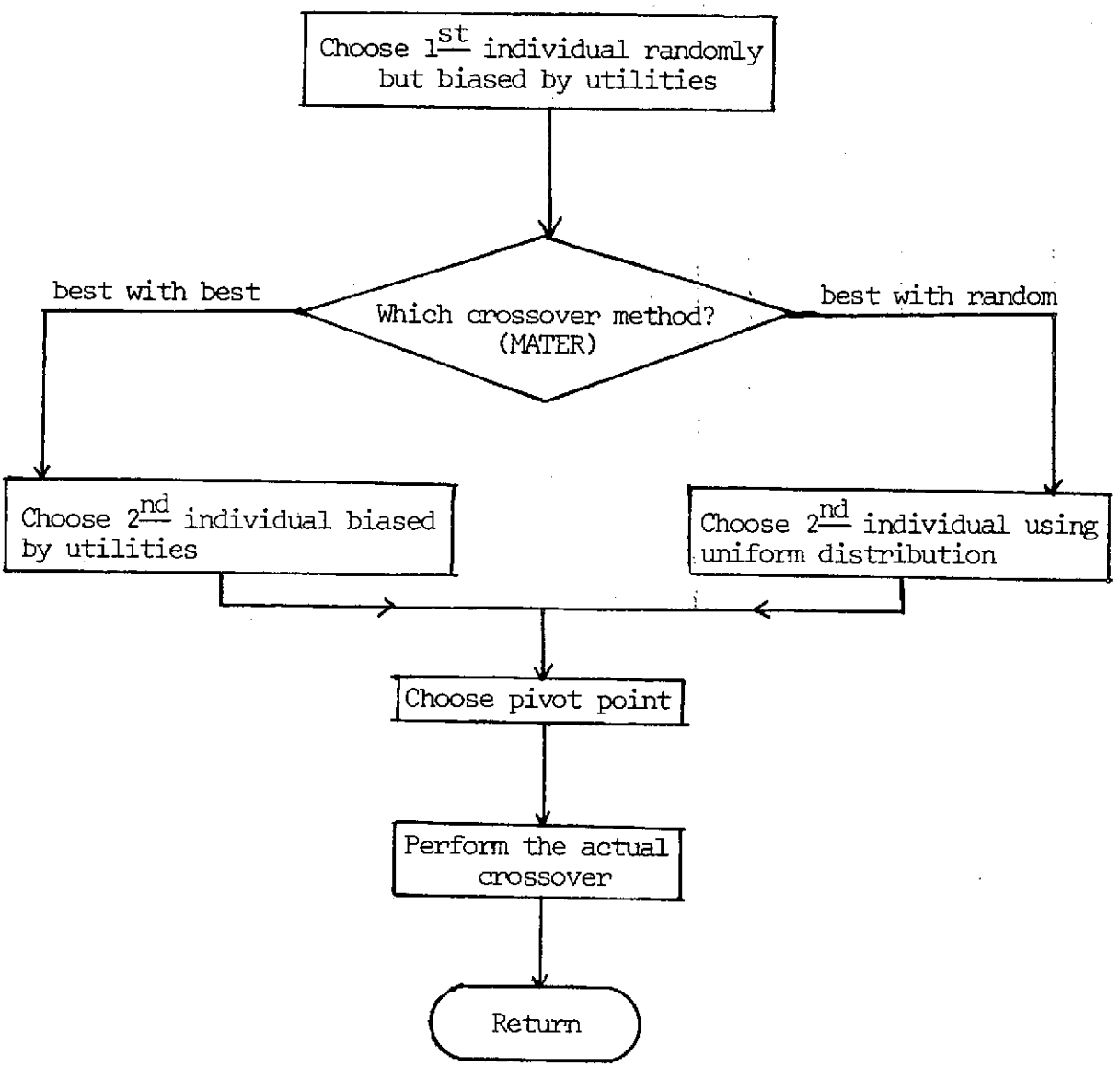
NALEL-1. The genetic operators work on these strings in the following ways.

Crossover:

Two strings are chosen from the current population. The first string is chosen randomly with the probability of selecting a particular individual being proportional to that individual's utility. The second individual is then chosen from the remaining population in one of two possible ways: 1) the second string may be selected using the same scheme as for the first, or 2) the second string may be chosen so that each of the remaining strings is equally likely to be picked. We call the first method "best with best" and the second method "best with random." The parameter MATER controls which method is used and is fixed for the entire run, but may be changed for different runs. (All of our parameters are set at the beginning of a run and must be held constant throughout that run.)

A crossover point (pivot point) is now chosen. If the string has length n (the parameter NCOOR holds this value), then there are n-1 possible pivot points (the interstices), each of which is equally likely to be chosen as the actual pivot point. The strings are then "broken" at the pivot point and recombined so that each of the two new strings consists of the starting portion of one of the original strings followed by the terminal segment of the other. For example, given the two strings (a,b,c,d,e) and (f,g,h,i,j) and choosing the pivot point to be between the third and fourth position, we get (a,b,c,i,j) and (f,g,h,d,e) after crossover. The two original individuals are retained in the population -- they are not destroyed by crossover. The two new individuals are added to the current population.

CROSSOVER

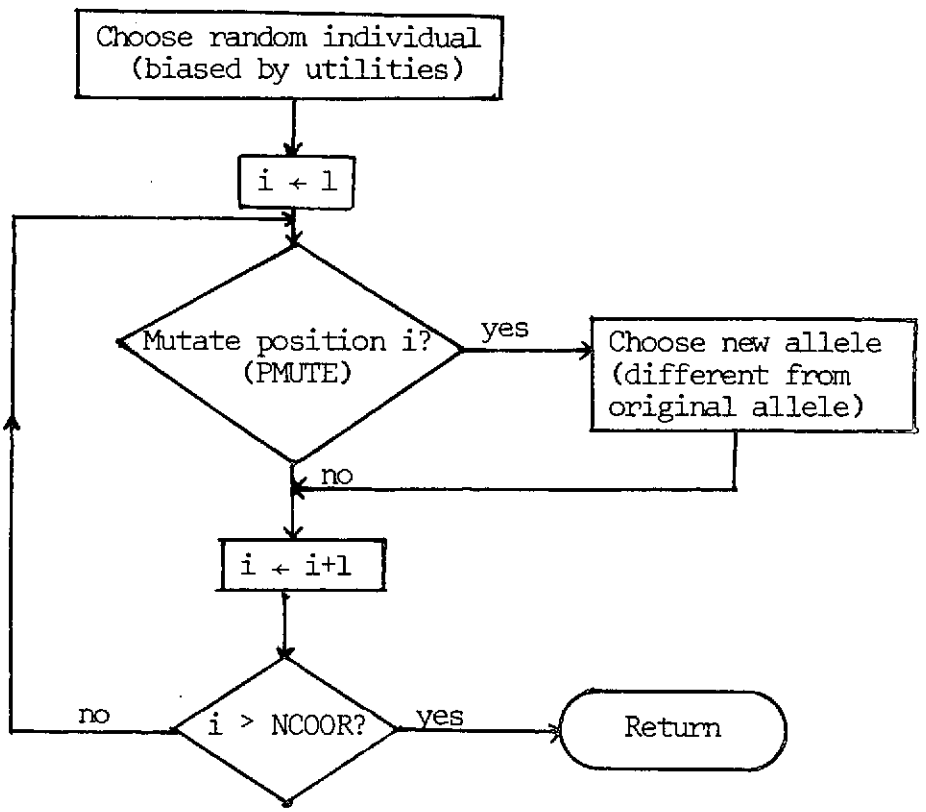

Choose 1st individual randomly but biased by utilities
Which crossover method? (MATER)
best with best
best with random
Choose 2nd individual biased by utilities
Choose 2nd individual using uniform distribution
Choose pivot point
Perform the actual crossover
Return

Mutation:

One individual is chosen from the current population. This choice is random, but biased according to utilities. For each position along the string, a decision is made whether or not to change the value at that position. The probability of making a change is controlled in our program by the parameter PMUTE and is fixed throughout a given run. If the value at the present position is to be changed, then each of the other possible values is equally likely to replace it. Again, the new individual is added to the population, and the original individual is also preserved.

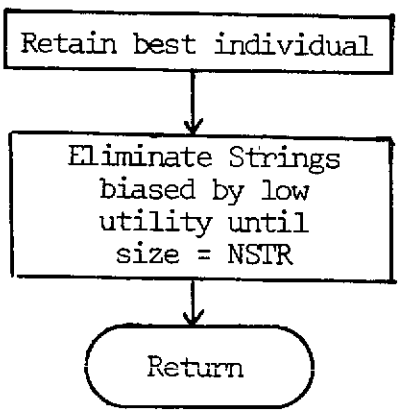
As an example, suppose that the string (k,l,m,n,o) has been chosen to undergo mutation. If PMUTE = .5, we have a 50-50 chance of changing k to a new value, 50-50 chance of changing l to a new value (independent of whether or not k has changed), etc. So the expected number of changes is 2.5 (PMUTE times the length of the string). The odds of making no changes are 1 in 32. The probability of changing all the coordinates is also 1/32. So the new string would most likely have 2 or 3 changes, say (k,l',m',n,o) or maybe (k',l,m',n',o). If the length of the string were 50, instead of 5, then the expected number of changes would be 25 and the chances of leaving the string unchanged would be only $(.5)^{50} \approx 10^{-15}$. On the other hand, if the length were 5 but PMUTE were .25, then the probability of changing k would be .25, the probability of changing l would be .25 also, etc. Now the expected number of changes would be 1.25 and the probability that the new individual is the same as the original individual is $(.75)^5 \approx .24$. So with PMUTE = .25, we would most likely find 1 coordinate changed, but quite often none or two would be changed.

MUTATION


Choose random individual (biased by utilities)
i ← 1
Mutate position i? (PMUTE)
yes
Choose new allele (different from original allele)
no
i ← i+1
i > NCOOR?
no
yes
Return

Finally, we followed the basic paradigm when reducing the population back to the original size, except that we always kept the best individual. The other individuals which survived the population reduction were chosen randomly with the probability of retaining a particular individual proportional to that individual's utility.

SELECTION



III. DESCRIPTION OF EXPERIMENTS AND RESULTS

We will use the term run to refer to a single execution of our program on the computer using one set of parameters and terminating when the stopping condition is satisfied. Experiment will mean a set of runs which use the same data and parameter values, except that each run uses a different random number seed. Since the algorithm is stochastic, we will usually refer to the values of different statistics for an experiment, meaning the average over the runs which make up that experiment, rather than referring to a particular run. As noted above, we used the "standard" function

$$f(x) = \sum_{i=1}^{NCOOR} ix_i$$

in all but a few of our experiments. In the remaining experiments, the objective function was Wood's function:

$$f(x) = 100(x_2-x_1^2)^2 + (1-x_1)^2 + 90(x_4-x_3^2)^2$$
$$+(1-x_3)^2 + 10.1((x_2-1)^2 + (x_4-1)^2)$$
$$+ 19.8(x_2-1)(x_4-1)$$

Actually, Wood's function was to be minimized rather than maximized, so we used -f(x) instead of f(x) and still maximized the objective function. Unless stated otherwise, the descriptions which follow are of experiments in which the "standard" function was to be maximized.

There are eight parameters which control our algorithm and which were varied from experiment to experiment:

1. NALEL -- the number of alleles (possible values) at each position. NALEL must be a positive integer greater than one (with a maximum value of about 15,000 on the computer we used).

2. NCOOR -- the number of dimensions of the domain space; also the length of the strings. NCOOR must be a positive integer not greater than 50.

4. MXSTR -- the maximum size of the population. When the population exceeds this size selection is immediately performed. MXSTR is an integer which should be at least as large as NSTR but no greater than 98.

5. XPROB -- the probability of choosing to use crossover rather than mutation during any particular time through the basic activity cycle.

6. MATER -- this parameter determines whether crossover is done on a best with best basis or a best with random basis.
   MATER = 1 for best with best
   MATER = 2 for best with random

7. PMUTE -- the probability of changing each coordinate of a string which is being mutated. PMUTE is a real number between 0 and 1 inclusive.

8. SPROB -- the probability of performing a selection, used each time through the main loop. SPROB is also a real number in the range 0 to 1.

The parameter space is quite large so we decided to restrict attention to the case of 2 alleles (NALEL = 2). We also limited ourselves to SPROB values of 1 and 0 -- either constant selection or else no selection until the population reached maximum size. We used 3 different initial population sizes: 10, 30, and 50 individuals. In these cases we chose maximum populations of 20, 60, and 98 individuals respectively. (98 was used since we had storage space for only 100 strings and a crossover could be performed when the population contained 99 strings, leading to 101 individuals before

the selection routine was called in.) With SPROB = 0 the population was thus allowed to approximately double in size before being cut back to the initial size. With SPROB = 1, the population was never more than 2 individuals larger then the initial population. Finally, we limited ourselves to optimizing the "standard" function on 10, 30, and 50 coordinates only.

Our first set of experiments consisted of using only crossover (XPROB = 1 -- no mutation -- and only using the best with best method for choosing the pairs to be operated on. There were 5 runs in each experiment and 18 experiments in this set (3 values of NCOOR, 3 pairs of NSTR, MXSTR values, 2 values of SPROB -- 3 x 3 x 2 = 18). The statistic values were plotted for each run and the deviation between the runs in an experiment was found to be fairly small (5 - 10%). In all of these experiments, saturation (as measured by statistics 3,4 and 5[1]) was rapid. The time

---

[1] Three measures of population variability were devised and recorded for each run. The first is simply the variance of the function value over the population. (It should be noted that although there are many different points with the same function value, we rarely observed more than 2 distinct points with the same function value in the population at any one time.) The second measure is a bit more complicated and is intended to measure the number of distinct individuals and so should compensate for the decrease in the variance of the function values caused because the points are close together even though distinct. This measure is given by the formula:

$$V = \frac{\left(\sum_{i=1}^{I} n_i\right)^{3/2} - \sum_{i=1}^{I} n_i^{3/2}}{\left(\sum_{i=1}^{I} n_i\right)^{3/2} - \sum_{i=1}^{I} n_i}$$

where I is the number of distinct points, and n_i are the number of occurrences of each of the I distinct points. For example, suppose there were 10 points, 4 of which take on one value, 3 take on another value, and the other 3 take on a third value. Then I=3, n_1=4, n_2=3 and n_3=3. The third measure is the average of the second measure applied to each coordinate over the coordinates. As it turned out, these three statistics were very similar and so we simply used the variance of the function value as our variability measure.

required before the process stagnates increases with population size, though not linearly, and is considerably shorter when selection was done constantly than when the population was allowed to double before selection. (See tables 1, 2, and 3, and figure 1.)

The population's performance, as measured by either the utility of the best individual or the average of the utilities of the entire population, improved as the population became larger. The rate of increase in performance was greatest at the beginning of each experiment (as one would expect) and the initial rate was greater for small populations than for large populations. The initial rate of improvement was also greater when constant selection was used rather than minimal selection. (See tables 1, 2 and 3, and figure 2.) It should be noted that the average utility is not a monotone increasing function of the number of function evaluations. Rather, the average utility seems to follow the best function value but lagging by a "random" amount (see figure 3). In fact, while the best value increases monotonically, it does not do so smoothly, but makes little jumps or occasionally larger jumps and remains constant between. All graphs given in this report are thus "idealizations" of the actual graphs obtained from the plotter.

We were also interested in the effect the mutation operator has on optimization of the function, and the role it plays in averting and/or delaying saturation. To this end, we ran three sets of 9 experiments, with XPROB = .5, .7, .9 (so that the probability of applying the mutation operator at any given time was .5, .3, and .1, respectively). The other parameters varied in each set of experiments were NCOOR (which took on values of 10, 30, 50) and PMUTE (which took on the values .1, .3, and .5). We applied the algorithm to an initial population of 10 strings which was allowed to double before selection was used (SPROB = 0). Since the variations between the runs in each of the previous experiments was rather low, we decided to try only 3 runs per experiment. Again the

variance between the runs which made up an experiment was quite low (5 - 10%), so we felt that 3 runs per experiment was probably sufficient.

In these experiments, the population maintained a large variability (as measured by any of statistics 3, 4, or 5) even after progress toward the optimum (statistic 1) had stopped. For a fixed value of XPROB, the variability of the population increases as PMUTE increases. And for a fixed value of PMUTE, the variability decreases as XPROB increases. (See table 4.) In terms of finding the optimal function value, we found that PMUTE = .1 was always the best setting for PMUTE (or at least as good as any other value of PMUTE), regardless of the values of the other parameters. With PMUTE = .5 performance was always markedly worse than with PMUTE = .1. The performance with PMUTE = .3 was sometimes as poor as with PMUTE = .5 but sometimes was not noticeably different from PMUTE = .1. (See tables 5, 6, and 7.) It should be noted that no matter what the value of PMUTE (or, for that matter, XPROB), a better function value is achieved when the mutation operator is used in conjunction with the crossover operator than when crossover is the only operator (compare tables 2 and 3 against tables 6, 7).

We also ran two sets of experiments using XPROB = .5, and PMUTE = .05 and .2 in an effort to discover if PMUTE = .1 was truly the optimal setting. The differences between these sets of experiments were rather small. Using PMUTE = .05 was slightly better on 30 coordinates than PMUTE = .1, but slightly worse on 50 coordinates. PMUTE = .2 was always slightly worse than PMUTE = .1. Thus it seems that PMUTE = .1 is a near-optimal setting in these cases, if not truly optimal.

We also investigated the effect of using constant selection with both operators. To this end, we ran 3 sets of experiments using XPROB = .5 in all of them and PMUTE = .1, .3, .5 in each set of experiments respectively.

These sets of experiments were compared to those made previously in which all parameters were the same except for the SPROB value. Using constant selection improves the performance with respect to both the final function value attained and the number of evaluations required to achieve the final value (except for one case, see table 12). This use of constant selection also decreases the population variability.

Our next series of experiments were made using only the mutation operator -- no crossover. We used values of .1, .3 and .5 for PMUTE. We used minimal selection (SPROB = 0) and populations of 10-20 and 30-60 strings. The standard function was optimized for 10, 30, and 50 coordinates. The experiments using PMUTE = .3 always reach their final function value quicker than those with PMUTE = .1. An equal or better final value is attained with PMUTE = .1. The only case where PMUTE = .3 seems better by both criteria is on 10 coordinates. (See tables 8, 9, and 10.) This is not surprising since the expected number of coordinates to be changed with PMUTE PMUTE = .1 is only 1. Hence, many strings which undergo mutation are not changed at all, thereby completely wasting one function evaluation (probability of no change is $.9^{10} \approx .35$ so about 35% of the "mutations" do nothing). With PMUTE = .3 it is much less frequent that a string is left unchanged ($.7^{10} \approx .028$, or about 3% of the time).

As our next set of experiments, we tried using the "best with random" type of crossover. We first tried crossover only with populations of 10-20 and 30-60 strings (SPROB = 0). Then we tried both operators (XPROB = .5, PMUTE = .1 and PMUTE = .3) on 10-20 and 30-60 strings. The performance and variability curves are almost identical to those of the corresponding experiments with the "best with best" type of crossover. It seems to make no difference whatsoever which type of crossover is used.

Our final group of experiments was done using Wood's function as the

objective function. Here we departed from the 2 allele case and allowed 400 values at each position. The possible values for each coordinate ranged from 0 to 4 in increments of .01 -- the optimum was (1,1,1,1). We used initial populations of 10 and 50 strings, allowing the population to double before selection. We tried mutation only with PMUTE = .5. We also tried crossover only, both the "best with best" and the "best with random" methods. Last, we tried both operators together (XPROB = .5, PMUTE = .5) using "best with best" in some experiments and "best with random" in others. The results are somewhat inconclusive. In terms of the final function value:

1. Large populations are better than small ones, except for mutation only experiments.
2. "Best with random" is usually, but not always better than "best with best".
3. The combination of mutation and crossover is usually better than either mutation only or crossover only.

See table 13 for more details. The performance curves for these experiments are not nearly so smooth as those for any of the experiments using the standard function. It is very difficult to talk about the rate of approach to the final value since the curves look like jagged staircases (see figure 4) and since the variance between runs was rather high.

The variability of the population does not seem to decrease very much using both operators or using only mutation, but it does decrease somewhat using only crossover. The variability fluctuates a great deal with mutation only and is relatively stable with both operators or with crossover only. The variability is somewhat lower with mutation only than with both operators but higher than for crossover only.

IV. ANALYSIS AND CONCLUSIONS

1) Throughout all the experiments, the progress curves appear to be roughly an exponential increase toward the final value of the form

$$V(n) = (V_0 - V_f)e^{-rn} + V_f \quad \text{where}$$

$V(n)$ is the mean value achieved after n sample evaluations

V_0 is the mean value at the beginning of a run

V_f is the mean value which appeared to be asymptotically being approached

r is the exponential decay factor

Thus the properties of most interest in the study of convergence viz. the final value converged to, and the rate of convergence could be simply described by the parameters V_f and r respectively according to our results.

2) Increasing the population size always increases V_f and decreases r. (See figures 5 and 10.)

3) The effect of using constant selection rather than minimal selection is usually to increase V_f but sometimes to decrease V_f (see figures 5 and 6). In all cases, the rate of convergence is increased, although not always significantly. The change in r may be due primarily to the fact that the mean population size (mean over time) is reduced by using constant selection. Also, it should be noted that the selection process tends to reduce the variability of the population by discarding those individuals which are much different from the best string since such individuals generally have low utility (see figure 7). So the use of constant selection should cause faster stagnation, which it does.

4) Using the crossover operator only -- without mutation -- gives progress curves which have relatively low values of V_f, but have relatively high

values of r ("relative" to using mutation only or using both operators together). (See figures 8 and 9.) The reason for the rapid decrease in population variability when using crossover only is probably due to the following. Early in the run, a string is created which is far superior to the remainder of the population. It is, thereafter, very likely that this string will be chosen as one of the two strings to be crossed over almost every time a crossover is done. The result of these crossovers is to introduce more strings which are similar to the best string (one of the crossover products always has more than half of its alleles the same as the best string). These strings have better function values than most of the population, so that selection tends to favor keeping many very similar, if not identical, strings. This effect then snowballs as the population becomes dominated by these very similar strings. As shown by Holland [see Holland 1973], the proportion of a given schema should increase exponentially at a rate proportional to its utility relative to the average utility of the population. This suggests that the schema represented by the best string in the initial population should come to dominate the population rather quickly and that progress must halt (unless new schema can be introduced by some other method, such as mutation) when that happens.

This analysis predicts that the "best with random" method should lead to a smaller r and less rapid decline in population variability. This does not seem to be the case. Indeed, looking at the experiments using Wood's function, we see that the variability is much lower using "best with random" than for "best with best" -- just the reverse of what one would expect. On the other hand, the value of V_f is better for "best with random". It should be noted that the runs on Wood's function were terminated before the variability dropped to zero and so we may be analyzing only the initial

portion of the progress curves.

The fact that crossover leads to rapid saturation of the population suggests an obvious change to the selection procedure -- keep only 1 copy of each string. If no duplicates are permitted, then the variability must remain high. Such a change would greatly increase the running time of the selection procedure (selection already consumes most of the running time of the optimization program) and would probably decrease the rate of convergence. It should, however, increase the final value obtained.

5) Using mutation only -- no crossover -- avoids the problem of saturation and keeps the variability fairly high. The rate, r, is generally lower than for crossover only (with PMUTE = .5 the rate is about the same). The same tradeoff occurs between V_f and r -- changing PMUTE increases one but decreases the other (see figure 11).

Intuitively one would expect that using very small values of PMUTE would lead to rather slow progress since the mutation process would often leave a string unchanged. Using very large values of PMUTE should lead to some rapid initial improvement followed by very slow progress (if any) because all new strings would be much different from the current strings and after getting about half the coordinates correct, no more progress could take place. Somewhere in between the progress should be reasonably quick and yet still give good final values. It seems that when the expected number of changes is rather small (on the order of 1-3), then the performance should be very good. The fact that PMUTE = .3 and PMUTE = .1 both achieve the same V_f on 10 coordinates, but that PMUTE = .1 is superior on 30 and 50 coordinates tends to confirm this hypothesis. Also, experiments with PMUTE = .5 are always markedly worse than those with PMUTE = .1.

Most of our results using only mutation can be explained on the basis of the following simple model. Let N be the number of coordiantes (the length

of a string). Assume that the strings in the population have the last M coordinates correct and the first N-M incorrect (in fact, the strings usually do have the last several positions correct and the first ones are relatively random -- since the function weights the last ones more heavily). Then if the expected number of changes is fairly small we can say that:

Probability of improvement ≈ prob. of a single change in N-M region and no change in M region

$$= (N-M)pq^{N-M-1} \cdot q^{M}$$

$$= (N-M)pq^{N-1}$$

where P = PMUTE, q = 1-PMUTE

So the expected time to M+1 correct coordinates is $\frac{1}{(N-M)pq^{N-1}}$

And the expected time to reach M correct coordinates starting from M_0 correct initially is

$$n_M = \sum_{M_0}^{M} \frac{1}{(N-M)pq^{N-1}} = \frac{N}{pq^{N-1}} \ell n\left(\frac{N-M_0}{N-M}\right)$$

And so the progress curve should be described by an equation of the form

$$M(n) = (M_0-N)e^{-npq^{N-1}}+N.$$

So the rate of progress should be given by pq^{N-1}. When N is larger than 3 or 4, q^{N-1} will determine the progress rate. So larger q should mean faster progress. In fact this is true -- see figure 11 -- the exponential rate of approach at the end of the run is greater for PMUTE = .1 (or q = .9) than for PMUTE = .3 (or q = .7). Also, larger N should mean slower rate of appraoch for the same value of PMUTE -- this effect is shown by figures 12 and 13.

Nearly all runs start with about N/2 correct coordinate values in the best few strings. If we view the mutation as operating on 2 portions of the string -- the good and the bad portions -- then we have, for each mutation, a sequence of N/2 Bernoulli trials on each portion of the string. The binomial probability distributions associated with the number of changes in each portion may be approximated by normal distributions with mean $\frac{Np}{2}$ and variance $\frac{Npq}{2}$ (again p = PMUTE). The net improvement is then the number of excess changes in the bad portion of the string and is approximately normally distributed with mean zero and variance Npq. Thus the expected mean rate of <u>improvement</u> (assuming selection throws out those offspring which are worse than their parents) is roughly $\sqrt{\frac{Npq}{2\pi}}$.

This suggests that the initial slope of the progress curves should increase as the number of coordinates increases, or as the product pq increases. Again, this is observed -- see figures 11 and 12. Note that pq = .09 for PMUTE = .1 and pq = .21 for PMUTE = .3, so the initial slope is higher when PMUTE = .3.

The fact that the different values of PMUTE lead to different final values is explained by observing that the expected time to make an improvement is given by $\frac{1}{(N-M)pq^{N-1}}$ where M is the number of correct coordinate values. As q increases (or PMUTE decreases), this time decreases. So if we assume that the progress will appear stopped to the experimenter when the time until the next improvement exceeds 10,000 evaluations, then we may compute V_f by

$$(N-V_f)pq^{N-1} \cong 10^{-4}$$

$$V_f \cong N - \frac{10^{-4}}{pq^{N-1}}$$

Thus V_f is expected to increase as q increases (or as PMUTE decreases). Again, this is the case -- see figure 11. Also note that V_f should decrease relative to N as N increases (for a fixed value of q) -- see figure 13, and tables 8, 9 and 10.

6) Using both operators together gives better V_f than either operator alone, especially for small populations. The value of r is lower than for either alone. The optimal combination of mutation and crossover seems to be half and half (XPROB = .5). The optimal combination gives r about equal to the r for mutation only but a better V_f. So it seems that combining the two operators is one way of beating the trade off between V_f and r. The curves obtained from using both operators are very similar to the curves for mutation alone, especially with large populations (see figures 8 and 9). Thus we may hypothesize that mutation is the dominant operator and that adding crossover, or increasing the population size, or using constant selection somehow improves the density of good points (those which yield an improvement) in the space being searched by the mutation operator. If we suppose that the actual number of coordinates which can be effectively tested is increased, then we have

$$M(n) = (M_0 - N_{ef})e^{-npq^{N_{ef}}} + N_{ef}$$

as the progress curve and using other operators makes N_{ef} larger. This should have the effect of increasing V_f since

$$V_f = N_{ef} - \frac{10^{-4}}{pq^{N_{ef}}}$$

and decreasing r since $r = pq^{N_{ef}}$. Hence we have a trade off between V_f and r.

It also seems that the mutation operator is the source of population variability. Based on this hypothesis, we predict that the variance in function value should be proportional to the product (1 - XPROB)· PMUTE. That is, the percentage of alleles undergoing mutation during each cycle should be the source of the variance in the function value. This does seem to be approximately correct -- see Table 11.

TABLES AND FIGURES

Crossover operator only -- no mutation

Standard function on 10 coordinates; optimal value is 165

Population Size	SPROB	Number of evaluations to reach final value of: function	variance	Final value of function (statistic 1)	Final variance (statistic 3)
10	1	100	180	155	0.
10-20	0	160	260	159	0.
30	1	220	980	163	0.
30-60	0	340	1820	164.6	0.
50	1	360	2180	165	0.
50-98	0	300	4500	165	0.

TABLE 1

Crossover operator only -- no mutation

Standard function on 30 coordinates; optimal value is 1395

Population Size	SPROB	Number of evaluations to reach final value of: function	variance	Final value of function (statistic 1)	Final variance (statistic 3)
10	1	120	200	1198	0.
10-20	0	220	420	1215	0.
30	1	620	1940	1308	0.
30-60	0	1500	4980	1319	0.
50	1	1220	3420	1356	0.
50-98	0	1680	9140	1338	0.

TABLE 2

Crossover operator only -- no mutation

Standard function on 50 coordinates; optimal value is 3825

Population Size	SPROB	Number of evaluations to reach final value of: function	variance	Final value of function (statistic 1)	Final variance (statistic 3)
10	1	150	275	3048	0.
10-20	0	200	520	3160	0.
30	1	1060	2500	3311	0.
30-60	0	1700	5160	3378	0.
50	1	2860	6540	3580	0.
50-98	0	3760	9500	3515	0.

TABLE 3

Crossover and Mutation operators

Standard function on 10 coordinates. Population of 10 strings, initially, allowed to double before selection.

XPROB	PMUTE	Initial deviation	Final deviation	Average deviation
1.0	---	17.5	0.00	0.71*
0.9	0.1	17.9	0.70	4.32
0.9	0.3	20.8	7.07	7.75
0.9	0.5	19.4	14.4	9.93
0.7	0.1	18.0	13.6	8.43
0.7	0.3	20.8	17.1	16.4
0.7	0.5	19.4	11.9	18.6
0.5	0.1	17.9	17.0	11.1
0.5	0.3	17.9	21.5	18.1
0.5	0.5	17.9	22.5	23.6
0.0	0.1	18.0	22.8	17.8
0.0	0.3	18.0	25.2	21.3

TABLE 4

Population variability vs. crossover and mutation rates

* This value is misleading. In the case of no mutation, the deviation quickly drops to zero (after 300 evaluations in this case) and remains there. The average, thus, depends on how many evaluations are made before stopping the experiment (4000 evaluations for the experiments in this table).

Crossover and Mutation operators

Standard function on 10 coordinates -- optimal value is 165. Population of 10 strings, initially, allowed to double before selection. "Best with best" crossover method.

XPROB	PMUTE	Final function value	Number of evaluations needed to reach: final value	98% of final value
1.0	---	159	160	160
0.9	0.1	165	267	167
0.9	0.3	165	467	400
0.9	0.5	165	333	333
0.7	0.1	165	233	200
0.7	0.3	165	267	267
0.7	0.5	165	233	200
0.5	0.1	165	167	167
0.5	0.3	165	200	167
0.5	0.5	165	200	133
0.0	0.1	165	267	233
0.0	0.3	165	500	200

TABLE 5

Performance as a function of the parameters XPROB and PMUTE

Crossover and Mutation operators

Standard function on 30 coordinates -- optimal value is 1395. Population of 10 strings, initially, allowed to double before selection. "Best with best" crossover method.

XPROB	PMUTE	Final function value	Number of evaluations needed to reach: final value	98% of final value
1.0	---	1215	220	220
0.9	0.1	1382	1967	967
0.9	0.3	1371	2533	1067
0.9	0.5	1351	2600	1767
0.7	0.1	1385	1867	1367
0.7	0.3	1382	2067	833
0.7	0.5	1327	1200	600
0.5	0.05	1395	1667	1400
0.5	0.1	1392	1933	1100
0.5	0.2	1389	2000	1167
0.5	0.3	1380	3600	3500
0.5	0.5	1378	5400	3433
0.1	0.1	1358	1500	767
0.1	0.3	1276	1767	767
0.1	0.5	1312	4733	2533
0.0	0.1	1377	1333	967
0.0	0.3	1277	1000	500
0.0	0.5	1244	633	633

TABLE 6

Performance as a function of the parameters XPROB and PMUTE

Crossover and Mutation operators

Standard function on 50 coordinates -- optimal value is 3825. Population of 10 strings, initially, allowed to double before selection. "Best with best" crossover method.

XPROB	PMUTE	Final function value	Number of evaluations needed to reach: final value	98% of final value
1.0	---	3160	200	200
0.9	0.1	3724	3733	2500
0.9	0.3	3703	3600	2000
0.9	0.5	3624	3533	2267
0.7	0.1	3755	2933	2400
0.7	0.3	3696	7200	4833
0.7	0.5	3677	4300	3233
0.5	0.05	3694	2067	1533
0.5	0.1	3767	2400	1333
0.5	0.2	3698	4733	1233
0.5	0.3	3534	1733	1500
0.5	0.5	3553	2333	1733
0.1	0.1	3516	2433	1033
0.1	0.3	3309	1667	767
0.1	0.5	3240	1700	800
0.0	0.1	3610	3867	1400
0.0	0.3	3315	733	633
0.0	0.5	3248	1933	700

TABLE 7

Performance as a function of the parameters XPROB and PMUTE.

Mutation operator only

Standard function on 10 coordinates -- optimal value is 165. Population size allowed to double between selections.

Population Size	PMUTE	Final function value	Number of evaluations needed to reach: final value	98% of final value
10-20	.1	165	267	233
10-20	.3	165	500	200
30-60	.1	165	533	467
30-60	.3	165	300	267

Population Size	PMUTE	Initial deviation	Final deviation	Average deviation
10-20	.1	18.0	22.8	17.8
10-20	.3	18.0	25.2	21.3
30-60	.1	17.1	19.5	17.3
30-60	.3	17.1	22.2	20.5

TABLE 8

Performance and population variability using Mutation only.

Mutation operator only.

Standard function on 30 coordinates -- optimal value is 1395. Population allowed to double in size between selections.

Population Size	PMUTE	Final function value	Number of evaluations needed to reach: final value	98% of final value
10-20	.1	1377	1333	967
10-20	.3	1277	1000	500
10-20	.5	1316	4733	2533
30-60	.1	1380	4000	3100
30-60	.3	1276	1533	1400

Population Size	PMUTE	Initial deviation	Final deviation	Average deviation
10-20	.1	90.2	163	121
10-20	.3	90.2	140	125
10-20	.5	82.2	137	123
30-60	.1	96.8	134	114
30-60	.3	96.9	103	106

TABLE 9

Performance and population variability using Mutation only.

Mutation operator only

Standard function on 50 coordinates -- optimal value is 3825. Population allowed to double in size between selections.

Population Size	PMUTE	Final function value	Number of evaluations needed to reach: final value	98% of final value
10-20	.1	3610	3867	1400
10-20	.3	3315	733	633
10-20	.5	3240	1700	800
30-60	.1	3699	6233	4200
30-60	.3	3338	4333	1067

Population Size	PMUTE	Initial deviation	Final deviation	Average deviation
10-20	.1	171	266	290
10-20	.3	171	245	274
10-20	.5	171	254	274
30-60	.1	189	290	276
30-60	.3	189	245	234

TABLE 10

Performance and population variability using Mutation only.

Crossover and Mutation operators

Standard function on 10 coordinates; Population of 10-20 strings.

1 - XPROB	PMUTE	Average deviation	average deviation / $\sqrt{(1-XPROB)\ (PMUTE)}$
.1	.1	4.32	43.2
.1	.3	7.75	44.6
.1	.5	9.93	43.2
.3	.1	8.43	48.5
.3	.3	16.4	54.6
.3	.5	18.6	48.0
.5	.1	11.1	49.5
.5	.3	18.1	46.6
.5	.5	23.6	47.2
1.0	.1	17.8	38.9
1.0	.3	21.3	56.2

TABLE 11

Population variability is roughly proportional to $\sqrt{(1-XPROB)\ (PMUTE)}$.

Crossover and Mutation operators

Population of 30 strings, initially, selection either constant or minimal.
XPROB = .5 throughout.

PMUTE	SPROB	Final function value	Average deviation	Time to reach final value
.1	1	165	9.8	367
.1	0	165	13.5	367
.3	1	165	15.2	267
.3	0	165	18.5	367
		Standard function on 10 coordinates.		
.1	1	1395	79.4	2767
.1	0	1391	95.2	5400
.3	1	1395	124	4333
.3	0	1389	123	7400
		Standard function on 30 coordinates.		
.1	1	3825	206	8467
.1	0	3793	232	11100
.3	1	3647	257	11167
.3	0	3736	313	8900
		Standard function on 50 coordinates.		

TABLE 12

Effect of selection with both operators.

Wood's Function

Optimal value is 0. 400 alleles. Minimal selection

Population Size	XPROB	PMUTE	MATER	Final function value	Final variance*
10-20	0	.5	--	- 2.17	182
50-98	0	.5	--	- 3.36	944
10-20	.5	.5	1	- 6.02	727
50-98	.5	.5	1	- 3.36	980
10-20	.5	.5	2	- .959	164
50-98	.5	.5	2	- .942	771
10-20	1.0	--	1	-16.1	393
50-98	1.0	--	1	- 6.60	1989
10-20	1.0	--	2	-18.0	123
50-98	1.0	--	2	- 2.94	844

TABLE 13

Performance and variability on Wood's function.

* The final variance is indicative of the average variance since the value of this statistic did not vary very much during an experiment.

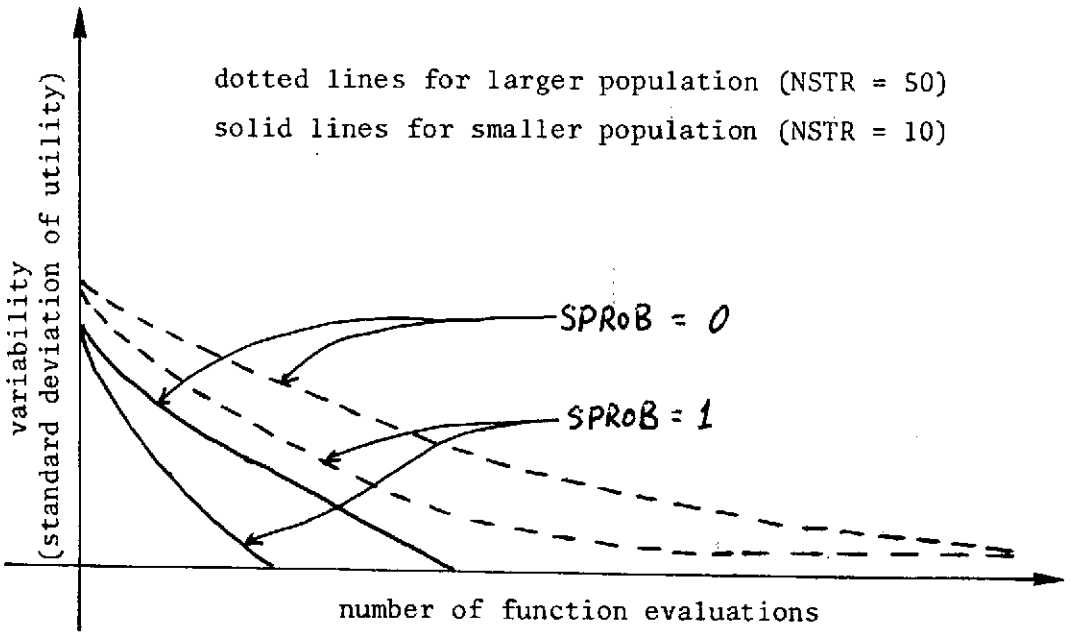


FIGURE 1

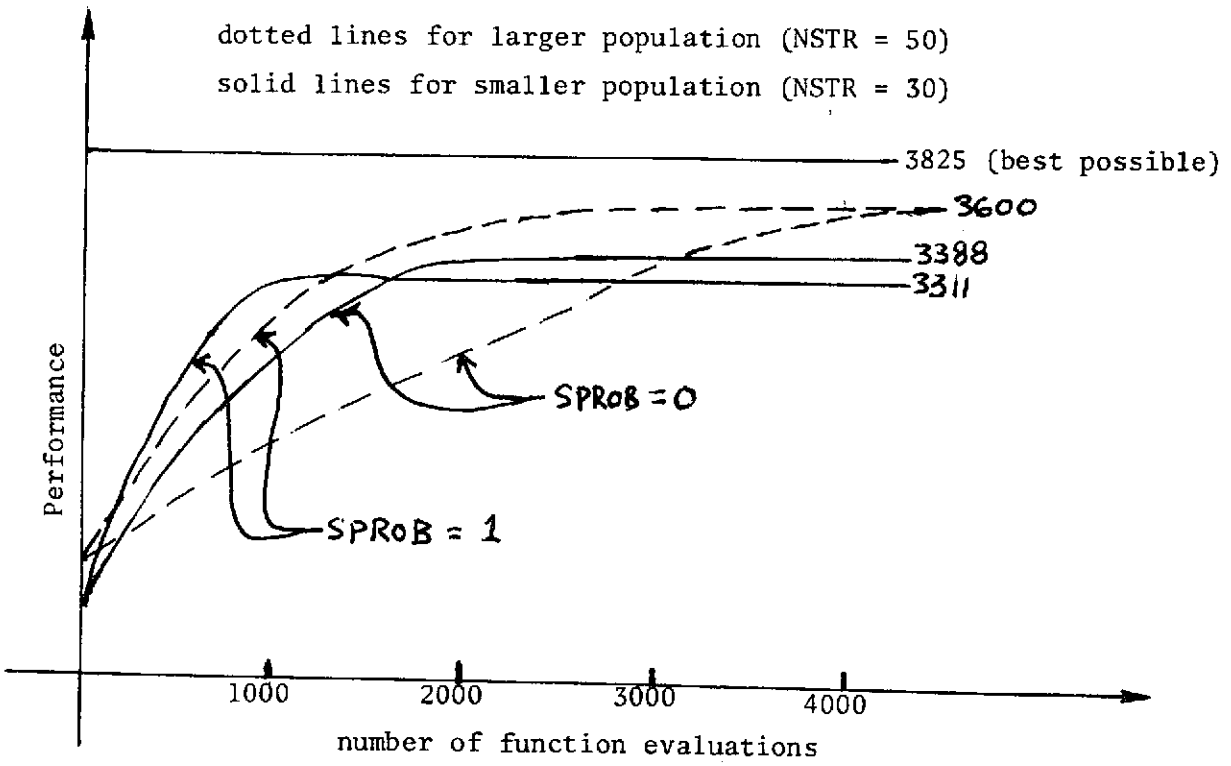


FIGURE 2

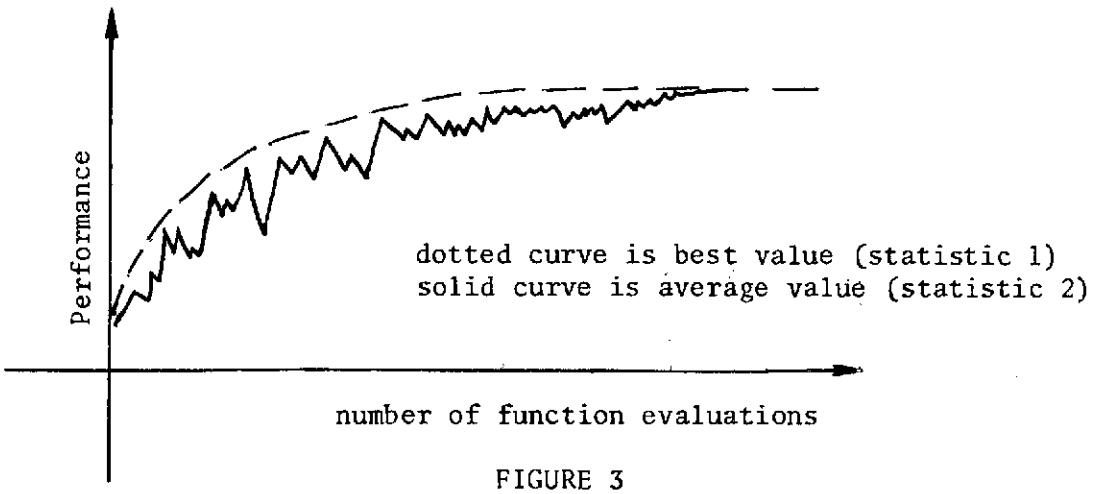


FIGURE 3

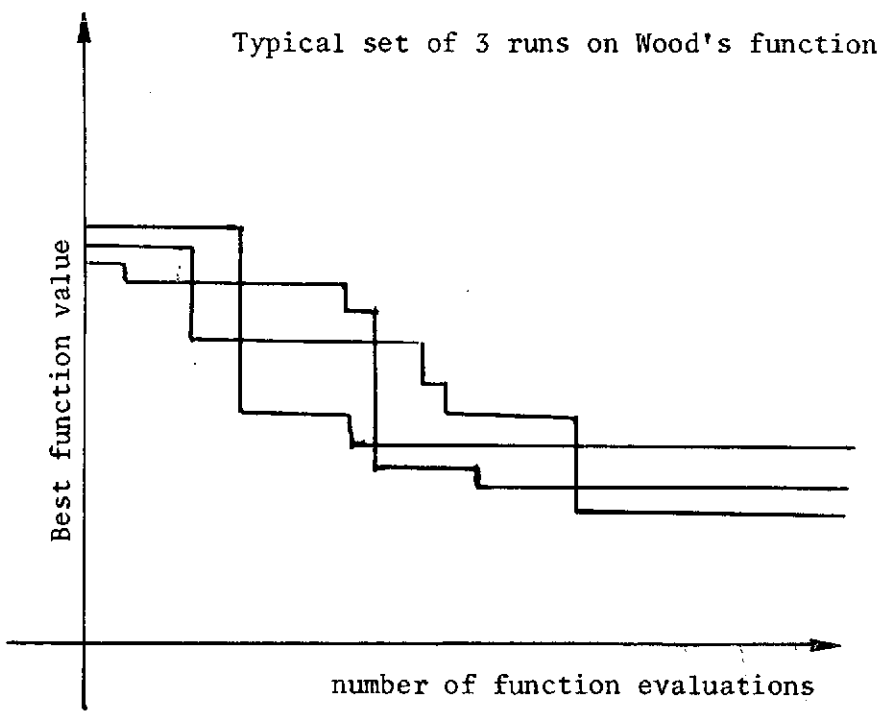
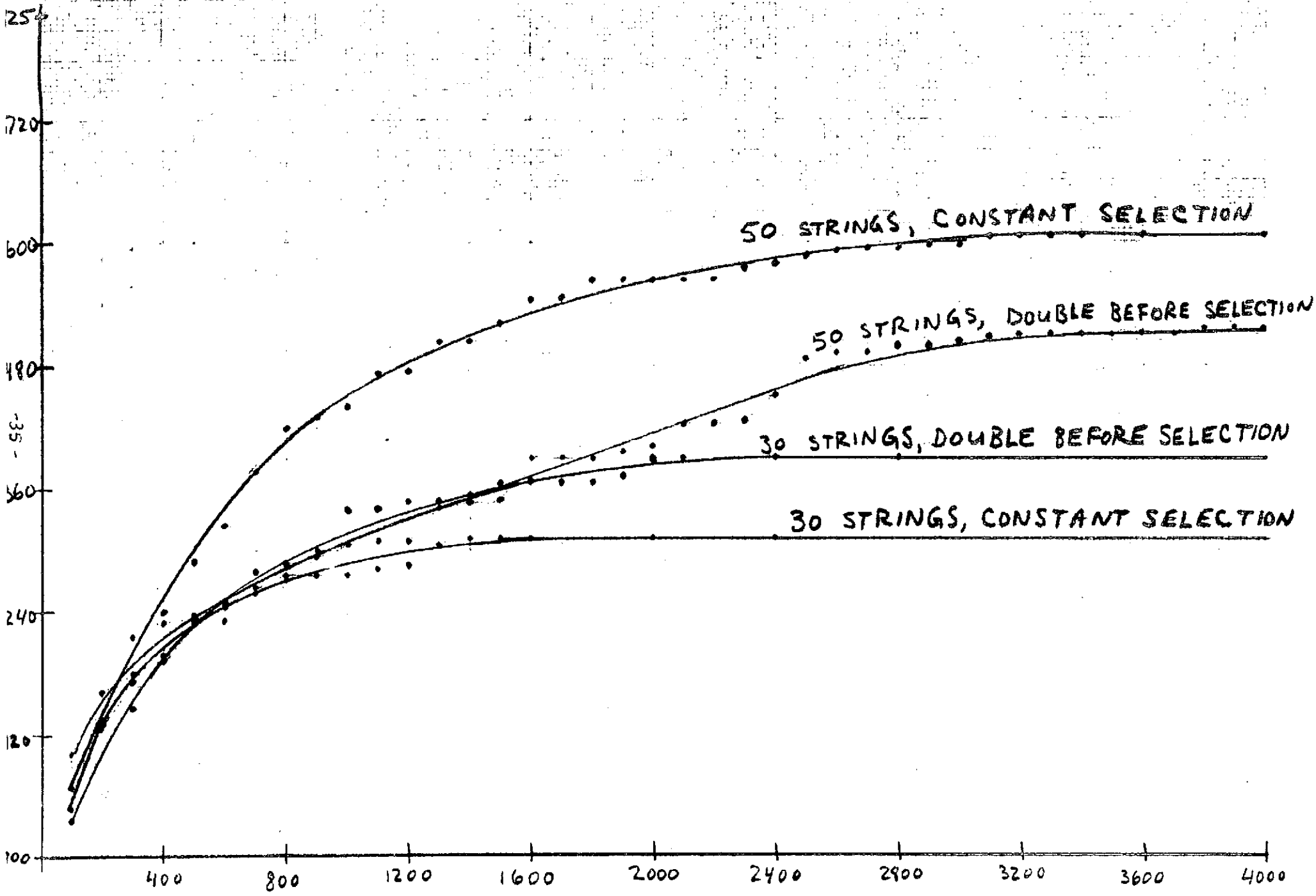


FIGURE 4

FIGURE 5

PERFORMANCE CURVES: STANDARD FUNCTION ON 50 COORDINATES



PERFORMANCE CURVES: PMUTE = .1

FIGURE 6

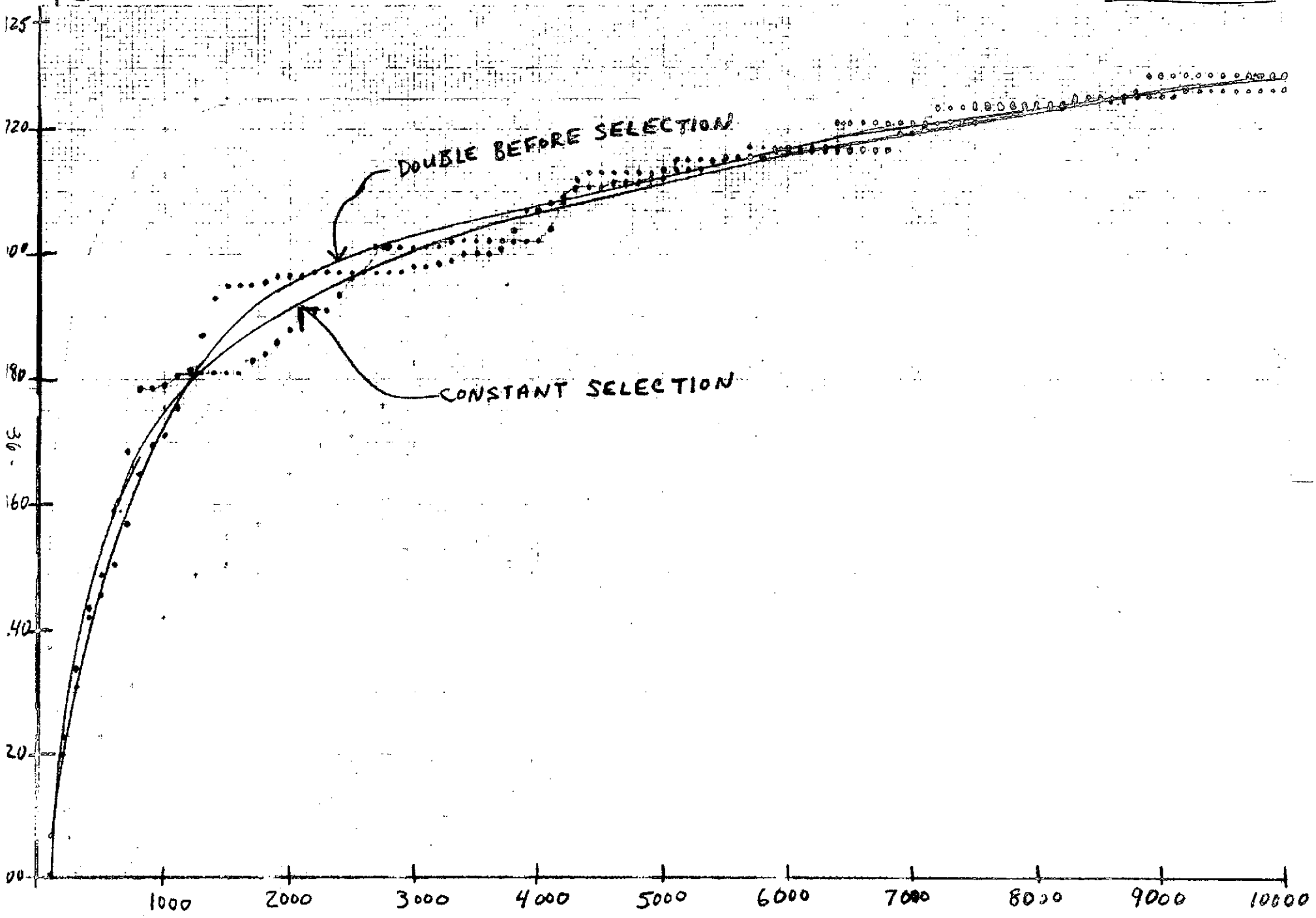
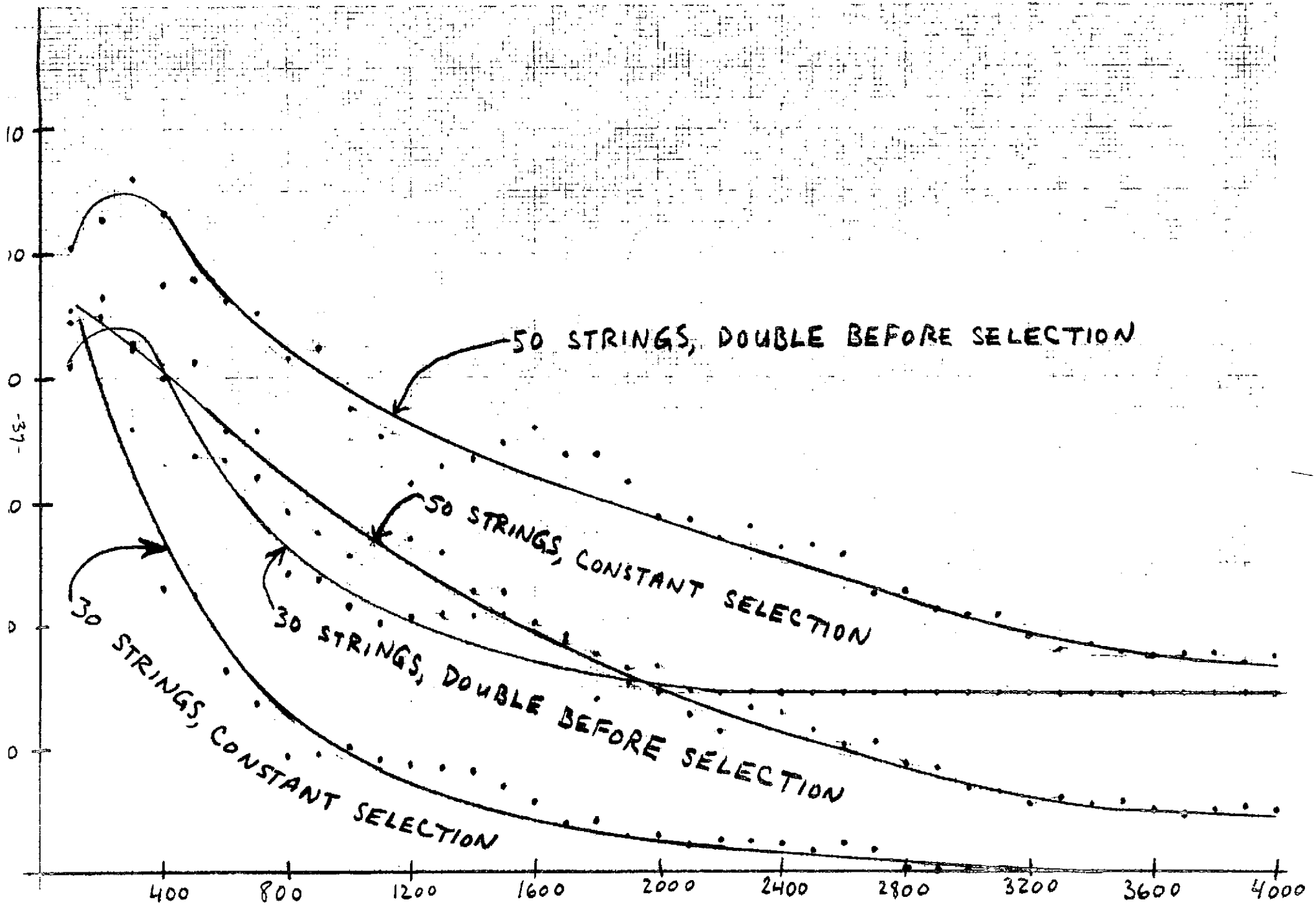
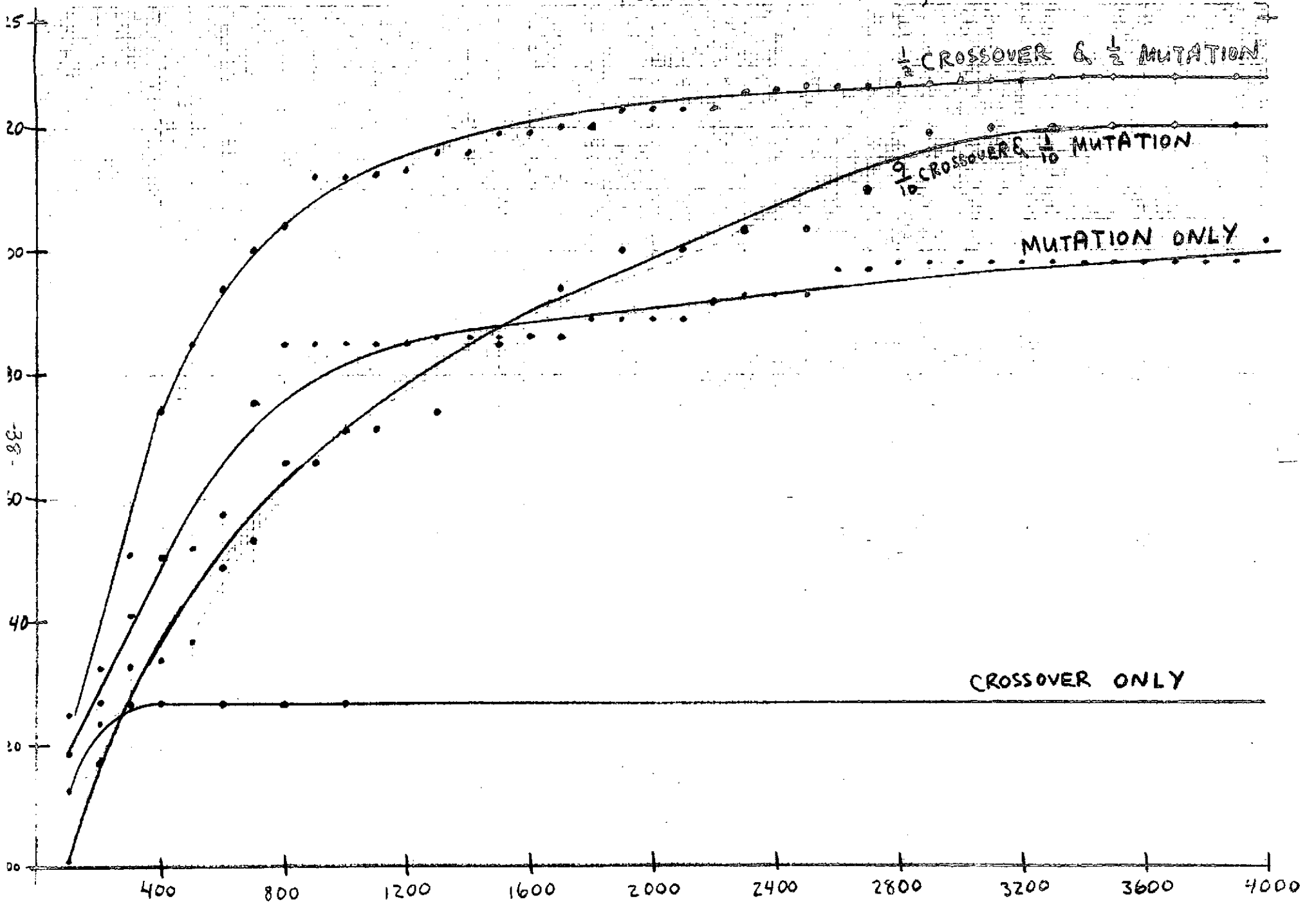


FIGURE 7

VARIABILITY CURVES: CROSSOVER ONLY.



PERFORMANCE CURVES — STANDARD FUNCTION ON 50 COORDINATES
POPULATION INITIALLY 10, ALLOWED TO DOUBLE BEFORE SELECTION



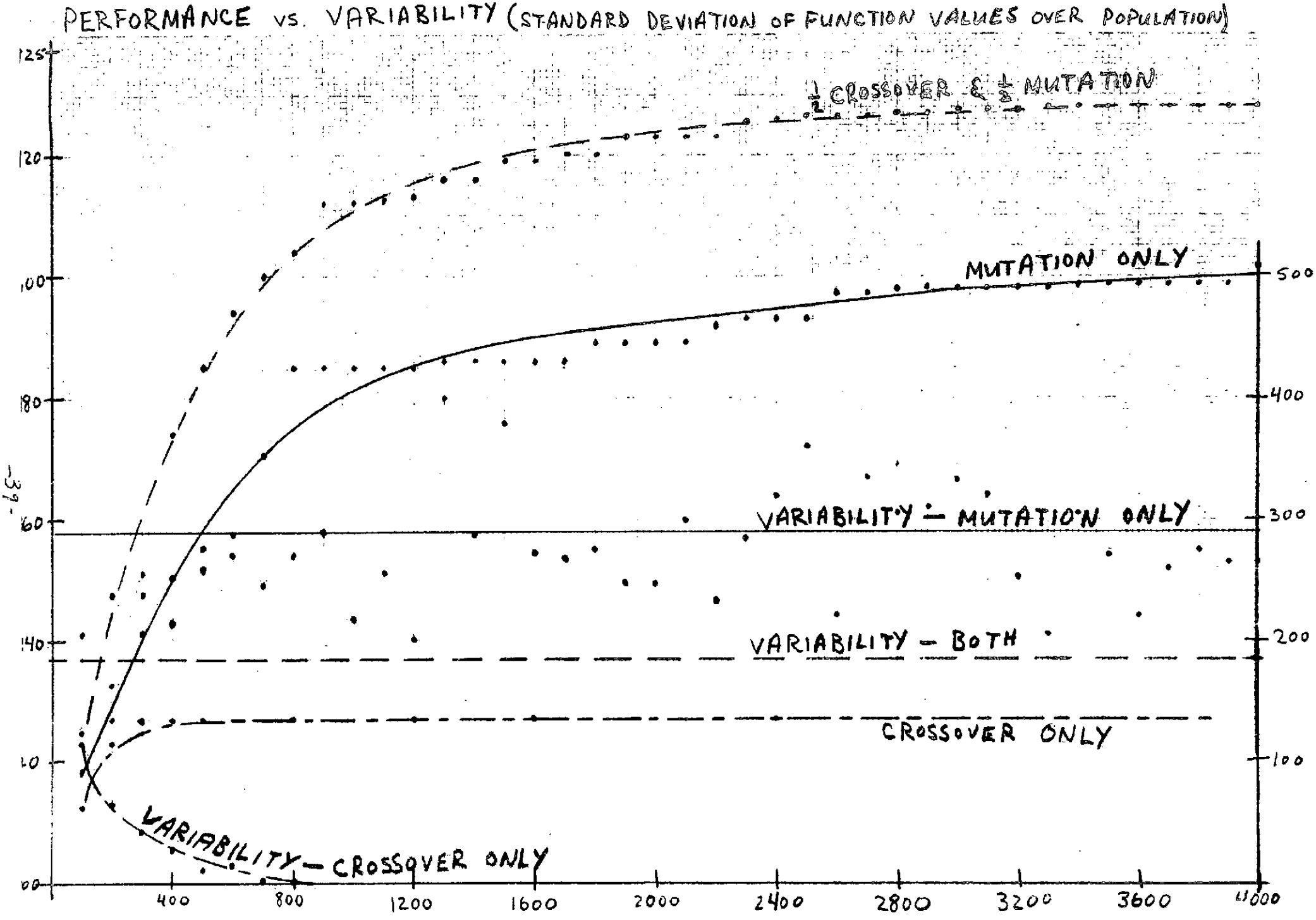
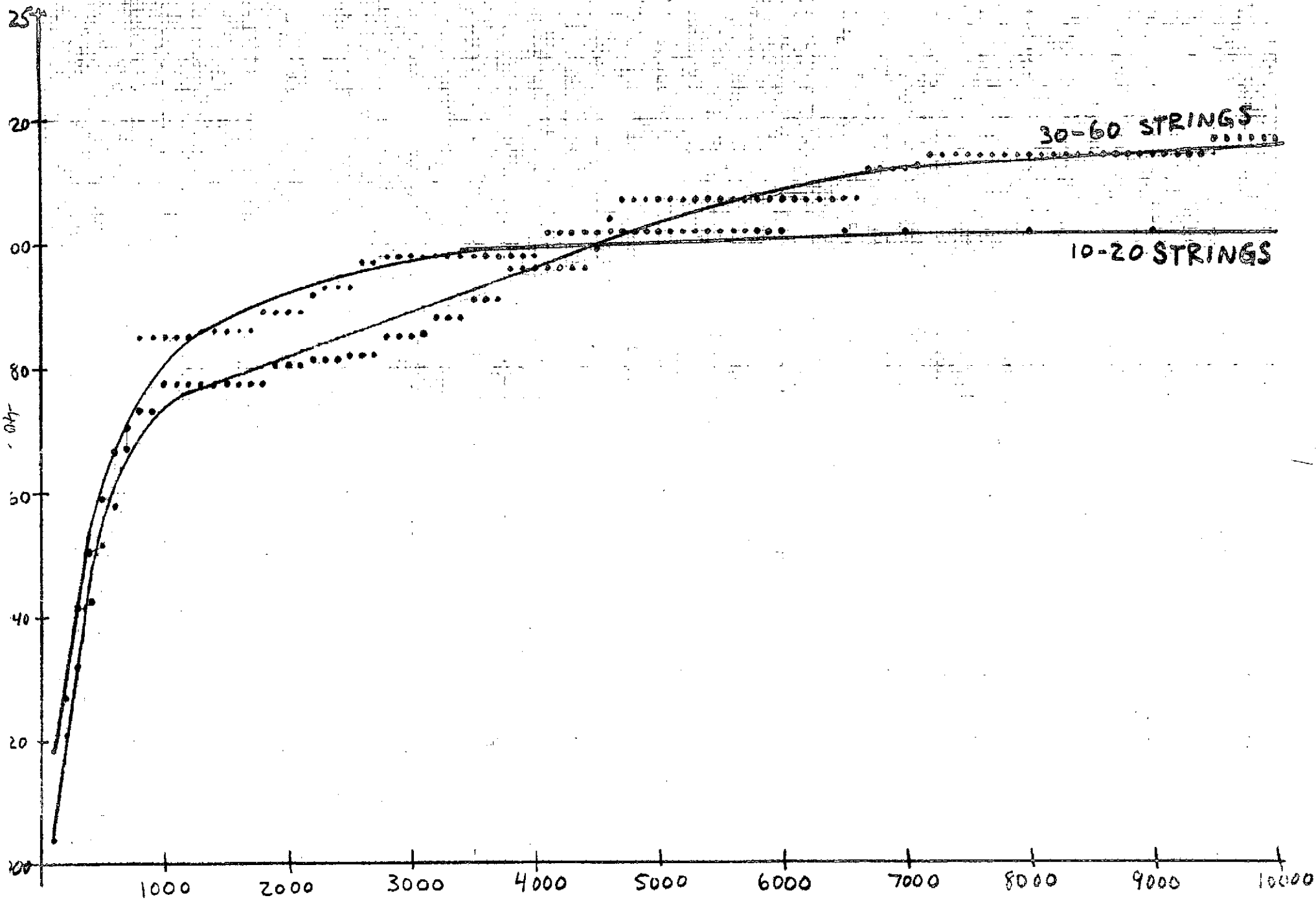
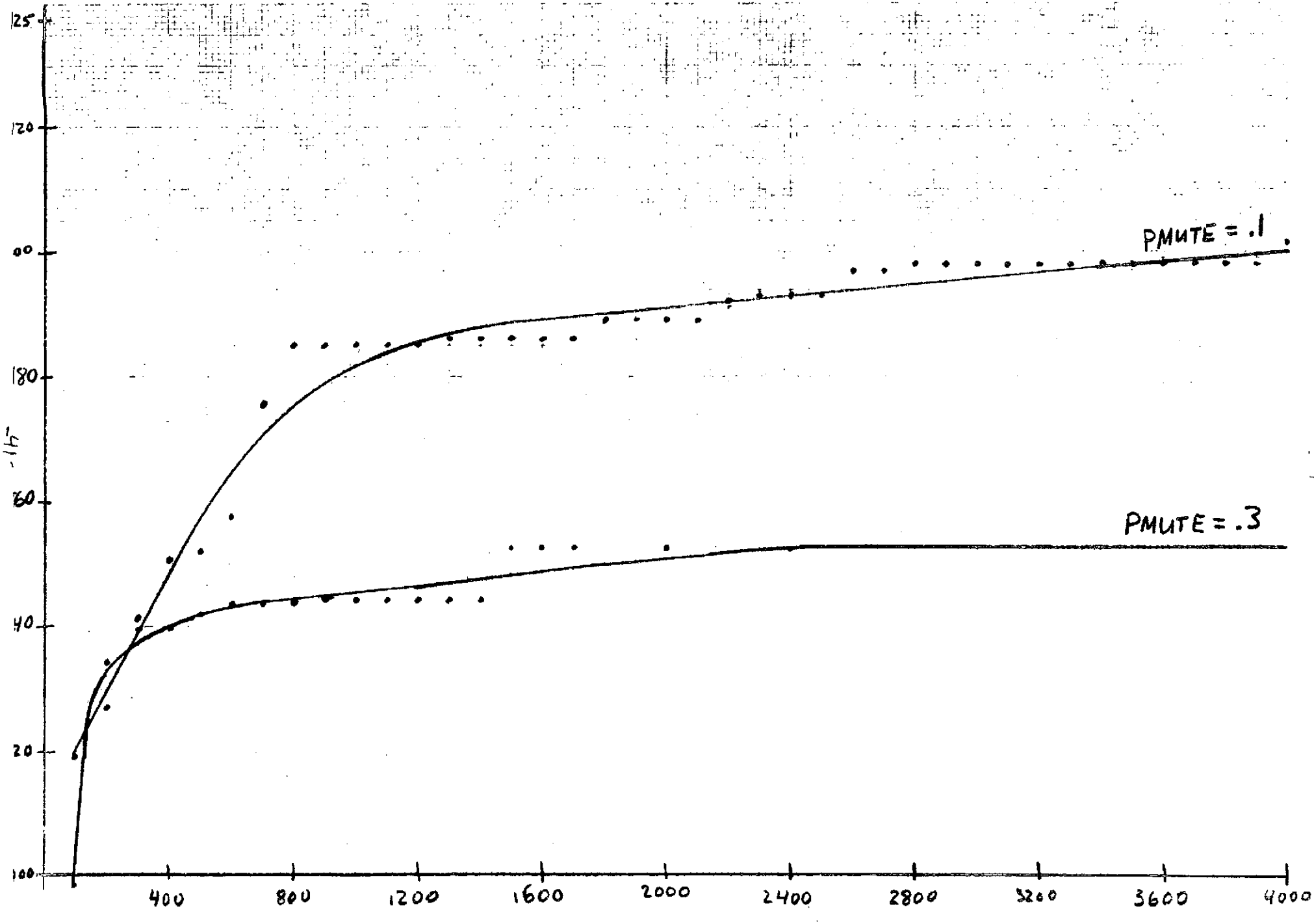

PERFORMANCE vs. VARIABILITY (STANDARD DEVIATION OF FUNCTION VALUES OVER POPULATION)
½ CROSSOVER & ½ MUTATION
MUTATION ONLY
VARIABILITY — MUTATION ONLY
VARIABILITY — BOTH
CROSSOVER ONLY
VARIABILITY — CROSSOVER ONLY
125
120
100
80
60
40
20
00
500
400
300
200
100
400
800
1200
1600
2000
2400
2800
3200
3600
4000

FIGURE 10

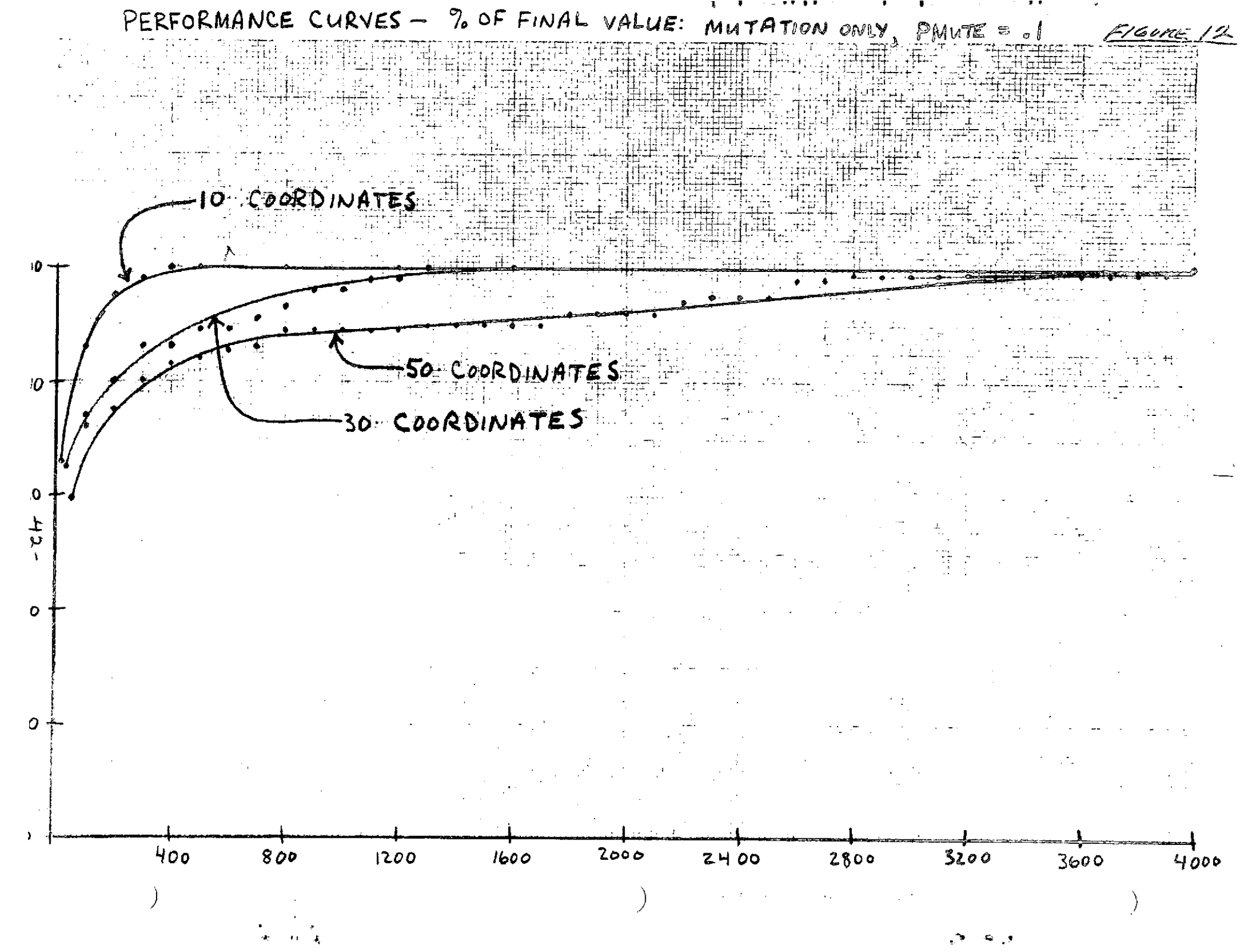
PERFORMANCE CURVES: STANDARD FUNCTION, 50 COORDINATES
MUTATION ONLY, PMUTE = .1



PERFORMANCE CURVES — STANDARD ... MUTATION ONLY. POPULATION OF 10 - 20



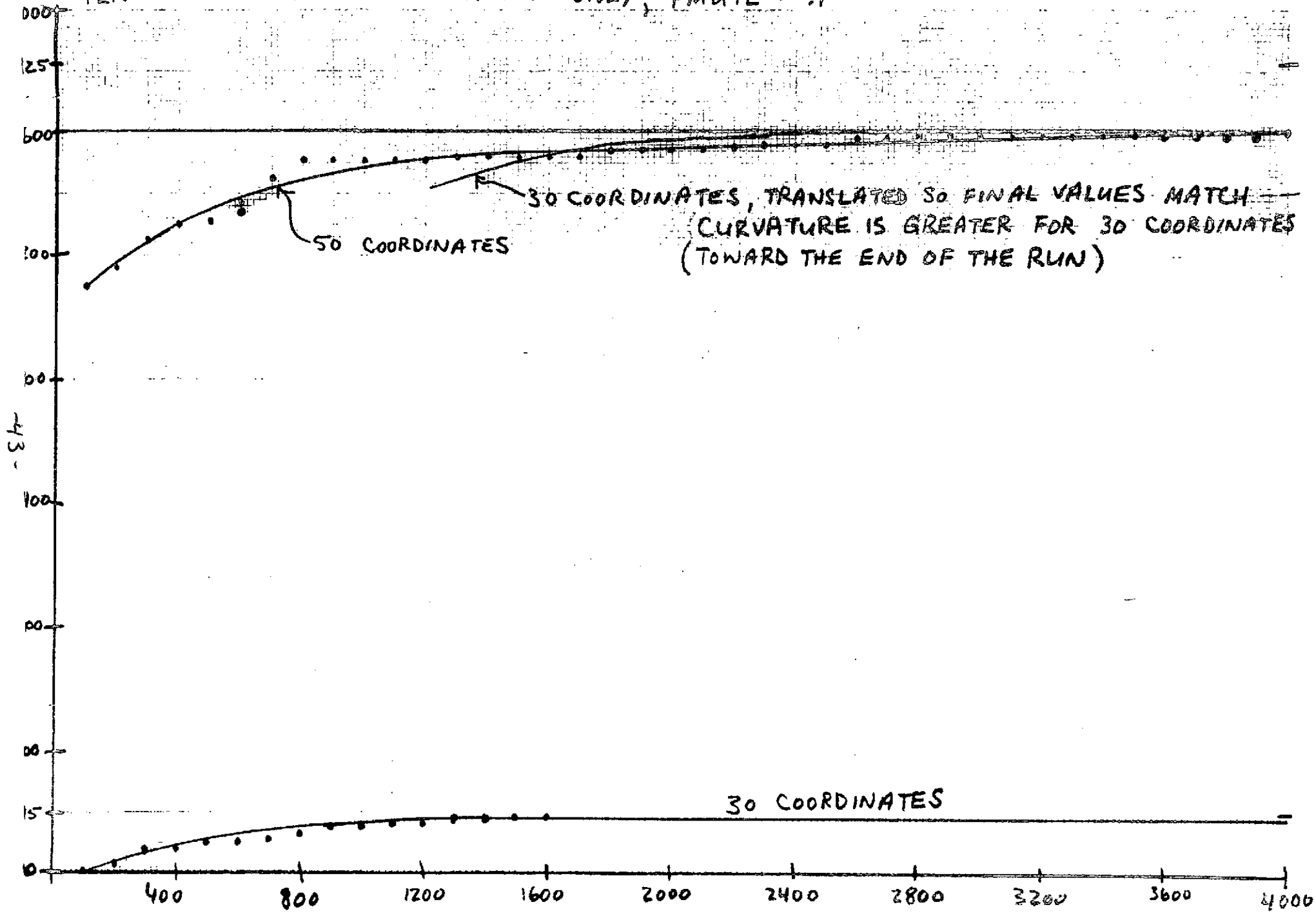
PERFORMANCE CURVES — % OF FINAL VALUE: MUTATION ONLY, PMUTE = .1 FIGURE 12



POPULATION OF 10-20 STRINGS

FIGURE 13

PERFORMANCE CURVES — MUTATION ONLY, PMUTE = .1



References:

Bosworth, J.L. and N.Y. Foo and B.P. Zeigler, "Comparison of Genetic Algorithms with Conjugate Gradient Methods", NASA Contractor Report No. 2093, NASA, Washington, D.C. August, 1972.

Foo, N.Y. and J.L. Bosworth, "Algebraic, Geometric and Stochastic Aspects of Genetic Operators", The University of Michigan, Computer & Communications Sciences Report No. 003120-2-T, March, 1972.

Holland, J.H., "Genetic Algorithms and the Optimal Allocation of Trials", SIAM J. Comput. June 1973.

Zeigler, B.P. and J.L. Bosworth and A.D. Bethke, "Noisy Function Optimization by Genetic Algorithms", The University of Michigan, Computer & Communications Sciences Report No. 143, March, 1973.